Scheduling Heuristics for Data Requests in an Oversubscribed Network with Priorities and Deadlines

Mitchell D. Theys[†], *Noah B. Beck*[†], *H. J. Siegel*[†], *Michael Jurczyk*[‡], and *Min Tan*[†]

[†]**Parallel Processing Laboratory**
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285 USA
{theys, noah, hj, min.tan.1}@purdue.edu

[‡]**Department of Computer Engineering and Computer Science**
University of Missouri - Columbia
Columbia, MO 65211 USA
mjurczyk@cecs.missouri.edu

Draft of Technical Report

September 1998

Abstract

Providing up-to-date information to users is an important data management problem for a heterogeneous networking environment, where each data storage location and intermediate machine may have specific data available, storage limitations, and communication links available. Sites in the network request data items and each request has an associated deadline and priority. This work concentrates on solving a basic version of the data staging problem in which all parameter values for the communication system and the data request information represent the best known information collected so far and stay fixed throughout the scheduling process. Three multiple-source shortest-path algorithm based heuristics for finding a near-optimal schedule of the communication steps for staging the data are presented. Each heuristic can be used with each of four cost criteria developed. Thus, twelve implementations are examined. The performance of the proposed heuristics are evaluated and compared by simulations. Many of the proposed heuristics perform very well with respect to a loose upper bound metric.

This research was supported by the DARPA/ISO BADD Program and the Office of Naval Research under grant number N00014-97-1-0804. M.D. Theys was also supported by an Intel Fellowship and an AFCEA Fellowship. Equipment used in this research was donated by Intel.

1. Introduction

The DARPA Battlefield Awareness and Data Dissemination (BADD) program [Roc96] includes designing an information system for forwarding (staging) data to proxy servers prior to their usage by a local application, using satellite and other communication links. The network combines terrestrial cable and fiber with commercial VSAT (very small aperture terminal) internet and commercial broadcast. This provides a unique basis for information management. It will allow web-based information access and linkage as well as server-to-server information linkage. The focus is on providing the ability to operate in a server-server-client environment to optimize information currency for many critical classes of information.

Data staging is an important data management problem that needs to be addressed by the BADD program. An informal description of an example of the data staging problem in a military application is as follows. A warfighter is in a remote location with a portable computer and needs data as input for a program that plans troop movements. The data can include detailed terrain maps, enemy locations, troop movements, and current weather predictions. The data will be available from Washington D.C., foreign military bases, and other data storage locations. Each location may have specific data available, storage limitations, and communication links. Also, each data request is associated with a specific deadline and priority. It is assumed that not all requests can be satisfied by their deadline. Data staging involves positioning data prior to its use in decision making for facilitating a faster access time when it is requested.

Positioning the data before it is requested can be complicated by the dynamic nature of data requests and network congestion; the limited storage space at certain sites; the limited bandwidth of links; the changing availability of links and data; the time constraints of the needed data; the priority of the needed data; and the determination of where to stage the data [Sma96]. Also, the associated garbage collection problem (i.e., determining which data will be deleted or reverse deployed to rear-sites from the forward-deployed units) arises when existing storage limitations become critical [Roc96, Sma96]. The storage situation becomes even more difficult when copies of data items are allowed to reside on different machines in the network so that there are more available sources from which the requesting sites can obtain certain data [TaS97], and so there is an increased level of fault tolerance, in cases of links or storage locations going off-line.

The simplified data staging problem addressed here requires a schedule for transmitting data between pairs of machines in the corresponding communication system for satisfying as many of the data requests as possible. Each machine in the system can be: (1) a source machine of initial data items, (2) an intermediate machine for storing data temporarily, and/or (3) a final destination machine that requests a specific data item.

All parameter values for the communication system and the data request information (e.g., requesting machines and network configuration) represent the best known information collected so far and stay fixed throughout the scheduling process. It is assumed that not all of the requests can be satisfied due to storage capacity and communication constraints. The model is designed to create a schedule for movement of data from the source of the data to a "staged" location for the data. It is assumed that the user can easily retrieve the data from this location. The full description of the model is presented in [TaT98]; portions are included to aid in the presentation of the heuristics detailed in this paper.

Three multiple-source shortest-path algorithm based heuristics for finding a near-optimal schedule of the communication steps for staging the data are presented. Each heuristic can be used with each of four cost criteria developed. Thus, twelve implementations are examined. The performance of the proposed heuristics are evaluated and compared by simulations. This research serves as a necessary step toward solving the more realistic and complicated version of the data staging problem involving fault tolerance, dynamic changes to the network configuration, *ad hoc* data requests, sensor-triggered data transfers, etc.

One of the heuristic/cost-criterion pairs was examined in [TaT98]. That investigation used a simpler model (e.g., there was no garbage collection), and a different set of network parameters (e.g., the number of machines that can connect to a machine).

Section 2 provides an overview of work that is related to the data staging problem. Section 3 presents the multiple-source shortest-path algorithm based heuristics for finding a near-optimal schedule of the communication steps for data staging. These heuristics adopt the simplified view of the data staging problem described by the mathematical model. A simulation study is discussed in Section 4, which evaluates the performance of the proposed heuristics outlined in Section 3. A BADD-like network environment has been used in developing the parameters for conducting this simulation study.

2. Related Work

To the best of the authors' knowledge, there is currently no other work presented in the open literature that addresses the data staging problem, designs a mathematical model to quantify it, or presents a heuristic for solving it. A problem that is, at a high level, remotely similar to data staging is the facility location problem [HuM89] in management science and operations research. Under the context of the construction of several new production facilities, a manufacturing firm needs to arrange the locations of the facilities and plants effectively, such that the total cost of transporting individual components from the inventory facilities to the manufacturing plants for assembly is minimized. It is required that the firm makes several interrelated decisions: how large and where should the plants be, what production method should be used, and where should the facilities be located? If an analogy is made between (1) the plants and the destination machines that make the data requests, (2) the individual manufacturing components and the requested data elements to be transferred, and (3) the production facilities and the source locations of requested data, then at a high level the facility location problem has features similar to those of the data staging problem (e.g., the use of a graph-based method to reduce the facility location problem to a shortest-path or minimum spanning tree problem).

However, when examining the relationship between the facility location problem and the data staging problem carefully, there are significant differences. First, each component that a plant requests is usually not associated with a prioritizing scheme, while in the data staging problem each data request has a priority. Also, each component request from a plant commonly does not have a corresponding deadline related factor, while in the data staging problem each data request has a deadline. For the data staging problem, the priority and deadline associated with each data request are the two most important parameters for formulating the optimization criterion. For example, the minimization of the sum of the weighted priorities of satisfiable data requests (based on their deadlines) is used as the optimization criterion in the mathematical model of a basic data staging problem presented in [TaT98]. But for the facility location problem, in general, researchers adopt optimization criteria that are related to the physical distances between plants and facilities in either a continuous or discrete domain without any prioritizing schemes or deadline related factors (e.g., [ChD81, CoN80, JoL95, MoC84, Shi77]). Thus, although lessons can be drawn from the design of algorithms for different versions of the facility location problem, there are no obvious direct correlations between either the formulations or the potential solutions of those two problems. In the facility location problem all constraints must be satisfied for the production to occur. In this research, it is known that not all requests can be satisfied.

Data management problems similar to data staging for the BADD program are studied for other communication systems. With the increasing popularity of the World Wide Web (WWW), the National Science Foundation (NSF) recently projected that new techniques for organizing cache memories and other buffering schemes are necessary to alleviate memory and network latency and to increase bandwidth [Bes97]. More advanced approaches of directory services, data replication, application-level naming, and multicasting are being studied to improve the speed and robustness of the WWW [BaB97]. Evidence has been shown that several file caches could reduce file transfer traffic, and hence the volume of traffic on the internet backbone [DaH93]. In addition, schemes for dynamic data replication have been created for distributed environments with the goal of increasing system performance by optimizing message traffic in the network [AcZ93]. The study of data staging can potentially draw lessons from and generate positive input for the active research in these related, but not directly comparable, areas of study.

Other research exploring heuristics for use in the BADD environment has been performed [LeB97]. This work examines methods for scheduling efficiently the ATM-like channels of a possible BADD-like environment. It shows that "greedy" heuristics are effective tools for use in that BADD-like environment and uses a network simulator to corroborate this statement; however, those heuristics do not consider several parameters considered here, such as deadlines and starting times. The work here differs from [LeB97] in that: (1) a mathematical model has been developed [TaT98], and (2) the collection of heuristics and cost criteria studied here are based on a different set of assumptions about system structure and data request characterizations.

3. Data Staging Heuristics

3.1. Introduction

The heuristics for solving the data staging problem are based on Dijkstra's algorithm for solving the multiple-source shortest-path problem on a weighted and directed graph [CoL90]. Dijkstra's algorithm takes as input a directed graph with weighted edges, and produces as output the shortest path from a set of machines to every other machine of the graph. More detailed information about Dijkstra's algorithm can be found in [CoL90]. The model used for the heuristics and implementation details about the heuristics is presented in Subsection 3.2. Subsections 3.3 through 3.5 present the heuristics that are investigated in this research. The four cost criteria developed for use with the heuristics are defined in Subsection 3.6. The mathematical model developed for the simplified data staging problem appears in [TaT98], where one of the twelve heuristic/cost-criterion pairs was examined using a simpler system model. Some information from that paper appears in this section as necessary background for the discussion of the data staging heuristics.

3.2. Heuristic Motivation

All necessary communication steps are scheduled by the data staging heuristics. The heuristics utilize the following three strategies collectively:

(i) Choose an order of data transfers that results in a set of data items being available on intermediate machines earlier. This set of data items will be chosen based on some criterion that will cause the ordering to satisfy more data requests.

(ii) Maximize the sum of the priorities of the potentially satisfiable data requests.

(iii) Consider the urgency of a request as its deadline approaches.

Assume there are $\underline{m}$ vertices in the graph $\underline{G}$ corresponding to the m machines in the network, $\underline{M[j]}$, $0 \leq j < m$. Suppose further that $\underline{V_S[i]}$ is the set of source vertices corresponding to the machines that are the initial locations of the data item $\underline{\delta[i]}$. Let $\underline{N\delta[i]}$ be the number of different machines where a request for $\delta[i]$ is initiated. Suppose $\underline{V_D[i]}$ is the set of destination vertices corresponding to the machines that are making data requests for $\delta[i]$. Let the length of a path from a source vertex $v_s \in V_S[i]$ to a destination vertex $v_d \in V_D[i]$ be defined as the difference between the time when data item $\delta[i]$ is available on v_s and the earliest possible time data item $\delta[i]$ arrives at v_d (via the machines and the communication links along the path). This time can be calculated using the various parameters defined in the model. With the above defined G, $V_S[i]$, and $V_D[i]$, a separate multiple-source shortest-path problem is well defined for each requested data item in the context of the data staging problem.

Dijkstra's algorithm, in general, is applied to a directed graph with weighted edges and a set of source vertices. The algorithm generates, in general, a shortest path from some source to every vertex in the network (using the weighted edges). For the heuristics examined here, Dijkstra's algorithm is executed on the directed network topology graph with weighted edges for each individual data item. The weights on any edge in the graph correspond to the communication time required to transfer the particular data item between the two vertices in the network. The set of source machines corresponds to $V_S[i]$. The implementation of Dijkstra's algorithm in the heuristics checks: (1) that all machines have enough memory capacity, Cap, to temporarily hold the the data item being transferred, (2) that the communication links are available, and (3) the initial time that the data item is available on a source. This information is for the shortest-path from some source vertex to a destination. At the completion of Dijkstra's algorithm, the shortest path from any source to all vertices in the network is known (clearly $V_D[i] \subseteq$ all machines). This shortest path time $\underline{A_T}$ is a lower bound for the actual transfer time when requests for all data items are considered collectively.

After applying the multiple-source shortest-path algorithm for each of the $\underline{\rho}$ requested data items $\delta[i]$ individually, ρ sets of shortest paths are generated. For the example shown in

Figure 1(a), there are four valid communication steps that can be scheduled (specified by asterisks). But different valid communication steps may have conflicting resource requirements (e.g., machine $M[0]$ cannot send data items $\delta[0]$ and $\delta[1]$ to machine $M[3]$ simultaneously due to a link conflict). Thus, a local optimization criterion is used to select one of the valid communication steps to be scheduled (refer to Subsection 3.6 for more information).

Readers should notice that it may be impossible to use the individually shortest paths to all destinations for each data item due to possible communication link and memory space contention in the network when transferring different data items during the same time interval. Also, a multiple-source shortest-path algorithm for G only attempts to minimize the time when a given requested data item is obtained by its corresponding requesting locations. But as clearly stated in Section 1, request deadlines and the priorities of the satisfiable data requests must be taken into account (as well as the sharing of the memory capacity of machines and the communication links by multiple data items).

The garbage collection scheme implemented here is now described. Leaving a copy of $\delta[i]$ on machine $M[r]$ after it is sent to the next machine on the shortest path allows this copy to be used as an intermediate copy for forwarding $\delta[i]$ to some other machines. For the example communication step of transferring $\delta[0]$ from $M[0]$ to $M[3]$ shown in Figure 1(a), by tracing the shortest-paths generated for $M[7]$, $M[8]$, and $M[9]$, the set of intermediate machines can be determined as $\{3, 5\}$. At some time duration, γ, after the last destination whose shortest path goes through $M[r]$ has received $\delta[i]$, the available memory capacity of $M[r]$ is incremented by $\mid \delta[i] \mid$ to simulate the removal of $\delta[i]$ from its memory. So for the example, $M[3]$ and $M[5]$ would keep $\delta[i]$ in their local memory for γ time units after the latest deadline among machines $M[7]$, $M[8]$, and $M[9]$. The data item is kept on the intermediate machines for this time duration to provide a level of fault tolerance in cases when communication links or storage locations become unavailable, or the case where a destination loses its copy of the data.

The next subsections discuss the three heuristics that have been developed. Subsection 3.6 presents background information about the cost criterion components, and details the four cost criteria used in this research.

3.3. Partial Path Heuristic

The heuristics presented in this section are built upon Dijkstra's multiple-source shortest-path algorithm. The heuristics iteratively pick which data item to transfer next based on a cost function. Each iteration of this heuristic involves: (a) performing Dijkstra's algorithm for each data request individually, (b) for the valid next communication steps, determining the "cost" to transfer a data item to its successor in the shortest path, (c) picking the lowest cost data request and transferring that data item to the successor machine (making this machine an additional source of that data item), (d) updating system parameters to reflect resources used in (c), and (e) repeating (a) through (d) until there are no more satisfiable requests in the system. In some cases, Dijkstra's algorithm would not need to be executed each iteration for a particular data transfer, i.e., if the data transfer did not use resources needed for any future transfers. This optimization is not yet considered because this research is not focusing on minimizing the execution time of the heuristics themselves.

This heuristic will schedule the transfer for the single "most important" request that must be transferred next, based on a cost criterion. The heuristic is called the partial path heuristic (referred to as partial in Figures 4 through 9) because only one successor machine in the path is scheduled at each iteration. The various cost criteria examined with this heuristic are described in Subsection 3.6, and this heuristic is evaluated in Section 4.

3.4. Full Path/One Destination Heuristic

The full path/one destination heuristic produces a communication schedule using fewer executions of Dijkstra's algorithm than the partial path heuristic. The behavior of the partial path heuristic showed that if a data item $\delta[i]$ was selected for scheduling a transfer to its next intermediate location (a "hop"), in the following iteration, the same requested data item, $\delta[i]$, would typically be selected again to schedule its next hop. The full path/one destination heuristic (referred to as full_one in Figures 4 through 9) attempted to exploit this trend by selecting a requested data item with one of the cost criterion discussed in Subsection 3.6 and scheduling all hops required for the data item to reach one of its final destinations before executing Dijkstra's algorithm again. Considering the example communication system in Figure 1(a), data item $\delta[0]$ would only be scheduled from $M[0]$ to $M[3]$ before executing Dijkstra's algorithm again in the partial path heuristic. In the full path/one destination heuristic, data item $\delta[0]$ would be scheduled from $M[0]$ to $M[9]$ (a destination) before executing Dijkstra's algorithm again. This results in reducing the number of executions of Dijkstra's algorithm by three for this example. A savings proportional to the average length of a data item's path from a source to a destination is expected from this heuristic. Considering again the communication system in Figure 1(a), if this heuristic initially schedules the transfer of data item $\delta[0]$ from $M[0]$ to $M[9]$, $M[3]$ and $M[5]$ would become sources for $\delta[0]$. In the next iteration, $M[7]$ could receive $\delta[0]$ from $M[5]$, and $M[8]$ could receive $\delta[0]$ from $M[3]$, without having to schedule a transfer from the original source, $M[0]$. This heuristic will schedule the entire path for the single "most important" request that should be transferred next. The schedules generated by the partial path and the full path/one destination heuristics are compared in Section 4.

The partial path heuristic may construct a partial path (of many links) that it later cannot complete (due to network or memory resources being consumed by other requested data items). However, until this is determined, the part of the path constructed may block the paths of the other requested data items, causing them to take less optimal paths or causing them to be deemed unsatisfiable (and removed from consideration). The full path/one destination heuristic avoids this problem. An advantage the partial path approach does have over the full path/one destination approach is that it allows the link-by-link assignment of each virtual link and each machine's memory capacity to be made based on the relative values of the cost criteria for the data items that may want the resource.

3.5. Full Path/All Destinations Heuristic

The full path/all destinations heuristic builds on the full path/one destination heuristic and requires fewer executions of Dijkstra's algorithm then the other two heuristics. In the full path/one destination heuristic, a data item is transferred from a single source to a single destination, even if there are multiple destinations requesting the same data item.

For the example communication system in Figure 1(a), $\delta[0]$ is requested by machines $M[7]$, $M[8]$, and $M[9]$, and the shortest path for these three destinations all originate at machine $M[0]$ and pass through machine $M[3]$. In the full path/all destinations heuristic (referred to as full_all in Figures 4 through 9), the data item $\delta[0]$ would be scheduled for all three destinations (machines $M[7]$, $M[8]$, and $M[9]$) at the same time. By scheduling the path to multiple destinations three fewer executions of Dijkstra's algorithm are required as compared to the full path/one destination heuristic. A savings proportional to the average number of destinations for a data item whose shortest path intermediate machine set share a common machine is expected. This heuristic will schedule all paths for a single data item that share the next machine in the path as an intermediate machine. This approach was considered because it was expected to generate results comparable to the full path one/destination heuristic, but with a smaller heuristic execution time. The schedules generated by the three heuristics are compared in in Section 4.

3.6. Cost criteria

Four cost criteria that use "urgency" and "effective priority" have been devised for the three heuristics presented. Each of these cost criteria was chosen so as to vary the effect these two parameters will have in determining the next communication step. This subsection begins by defining "urgency" and "effective priority" and the the four cost criteria used are then presented.

Suppose $A_T[i,j]$ denotes a lower bound on the time when $\delta[i]$ is received and available at its corresponding j-th requesting location (as mentioned in Subsection 3.2) and $Rft[i,j]$ denotes the finishing time (or deadline) after which the data item $\delta[j]$ on its k-th requesting location is no longer useful. Assume $M[r]$ is the next machine in the shortest path from a source to the j-th destination to receive data item $\delta[i]$. A satisfiability function $Sat[i,r](j)$, is 1, if a request for data item $\delta[i]$ is scheduled to be received at the j-th requesting destination before its deadline; and 0, if the request for data item $\delta[i]$ is scheduled to be received after the deadline on the j-th requesting destination. Note that if the request cannot be satisfied using the shortest path, there is no other path that will cause it to be satisfied. The set of destinations whose shortest paths for data item $\delta[i]$ all pass through the machine $M[r]$ is defined as $D\delta[i,r]$. As an example of the definition of $Sat[i,r](j)$, consider the shortest paths generated by selecting first the valid communication step for transferring $\delta[0]$ from $M[0]$ to $M[3]$ in Figure 1(a). For this example we only consider the vertices in $D\delta[i,r]$. Suppose that the request deadlines for $\delta[0]$ are as follows (in some abstract time units): $M[7]$, 10; $M[8]$, 15; and $M[9]$, 5. Suppose further that the shortest path estimate has shown that the network can deliver $\delta[0]$ at time: 12 for $M[7]$; 11 for $M[8]$; and 8 for $M[9]$. Then, $Sat[0,3](0) = 0$, $Sat[0,3](1) = 1$, and $Sat[0,3](2) = 0$

Suppose the priority of each data request is between 0 and P, where P is the highest priority possible (i.e., a member of the class of most important requests). $Priority[j,k]$ denotes the priority for the data request for the data item $\delta[j]$ on its k-th requesting location. Let $W[i]$ $(0 \leq i \leq P)$ denote the relative weight of the i-th priority. These weightings allow system administrators to specify the relative importance of priority α data request versus priority β data request, where $0 \leq \alpha, \beta \leq P$. Let $Efp[i,j]$ denote the effective priority for the data request of $\delta[i]$ from its j-th requesting location, where $Efp[i,j] =$

$Sat[i,r](j) * W[Priority[i,j]]$. Suppose $\underline{Urgency[i,j]}$ denotes the urgency for the data request of $\delta[i]$ from its j-th requesting location, where $Urgency[i,j] = -Sat[i,r](j) * (Rft[i,j] - A_T[i,j])$, where smaller $Urgency[i,j]$ implies that it is less urgent to transfer $\delta[i]$ to the j-th requesting location. The unit of measure for the $Urgency$ term is seconds. Suppose $\underline{W_E} \geq 0$ is the relative weight for the effective priority factor and $\underline{W_U} \geq 0$ is the relative weight for the urgency factor in the scheduling. Readers should notice that (a) applying Dijkstra's algorithm to obtain $A_T[i,r]$ through shortest paths, (b) maximizing $Efp[i,j]$, and (c) maximizing $Urgency[i,j]$ follow the three strategies for designing data relocation heuristics recommended in (i), (ii), and (iii), respectively, at the beginning of Subsection 3.2. For the valid communication step for transferring $\delta[0]$ from $M[0]$ to $M[3]$ shown in Figure 1(a), $Efp[0,0] = W[Priority[0,0]]$ and $Urgency[0,0] = -(Rft[0,0]-$ the time $\delta[0]$ is available at $M[0]$). Four cost functions for transferring the requested data item $\delta[i]$ from machine $M[s]$ to $M[r]$ via an available link are each defined using "urgency" and "effective priority".

If $Sat[i,r](j)$ is 0 for all r that correspond to valid next machines that receive $\delta[i]$ and the associated values of j, that request receives no resources and the data does not move from its current locations. The request is not eliminated from the network. Currently the heuristics are applied to a static system, as this constraint is loosened and a dynamic system is explored, links might become available that would facilitate the delivery of an otherwise unsatisfiable request. So requests that are at one point in time unsatisfiable, might become satisfiable at a later point in time.

The system has the following information: bandwidth available on a link, durations a link is available, and size of each data item. In the model used here, the time interval a link is needed to transfer a specific data item $\delta[i]$ is determined by dividing the size of the data item, $|\delta[i]|$ by the bandwidth available on the link. $\underline{L[i,j][k]}$ denotes the k-th direct <u>virtual communication link</u> from $M[i]$ to $M[j]$, where each $L[i,j][k]$ is associated with one unique time frame during which the corresponding physical link is available for communication.

Suppose that the current chosen communication step is to transfer the requested data item $\delta[i]$ from $M[s]$ to $M[r]$. Before repeating the above heuristics for determining the next communication step(s), the following information must be updated: the link availability times for any links that the heuristic has scheduled, the available memory capacity on any machines that $\delta[i]$ has been placed, the sources of $\delta[i]$ must now include all machines that $\delta[i]$ has been moved to/through, and the time at which $\delta[i]$ can be removed from any intermediate machines.

The cost criteria are designed so that the next chosen communication step should be the one that has the smallest associated cost among all valid next communication steps for transferring all $\delta[i]$ where $0 \leq i < \rho$. For the first cost criterion (<u>C1</u>), the $\underline{Cost_1[s,r][i,j][k]}$, for transferring the requested data item $\delta[i]$ from machine $M[s]$ to $M[r]$, via link $L[s,r][k]$, for the j-th destination, is defined as:

$$Cost_1[s,r][i,j][k] = -W_E * Efp[i,j] - W_U * Urgency[i,j].$$

The rationale for choosing the above cost for local optimization is as follows. First, only a valid next communication step whose associated $Sat[i,r]$ is not 0 will facilitate satisfying data request(s). The first term of $Cost_1[s,r][i,j][k]$ attempts to give preference to a satisfiable data request with a priority higher than the other requests. Furthermore, in order to satisfy as many data requests as possible, intuitively it is necessary to transfer a specific data item to the requesting locations whose deadlines are close to expire. This intuition is captured by the inclusion of the urgency term. Thus, collectively with the consideration of the priority of satisfiable data requests and the urgency of those data requests in this local optimization step, using this cost criterion in the data staging heuristic should generate a near-optimal communication schedule that reasonably achieves the global optimization criterion.

The second criterion (C2) examines the $Cost_2[s,r][i][k]$ for transferring the requested data item $\delta[i]$ from machine $M[s]$ to $M[r]$ via link $L[s,r][k]$:

$$Cost_2[s,r][i][k] = -W_E * \sum_{j \in D\delta[i,r]} Efp[i,j] + W_U * \min_{j \in D\delta[i,r]} -Urgency[i,j].$$

This cost function considers all requests for $\delta[i]$ whose shortest path passes through machine $M[r]$ and sums their weighted priorities. Rather than summing all of the urgency terms for the destinations, the most urgent satisfiable request is added in $Cost_2$. This method of capturing the urgency is used as a heuristic to maximize the sum of the weighted priorities of satisfied requests. (If the most urgent request for an item passing through $M[r]$ is satisfied, it is more likely that all requests for this data item passing through $M[r]$ will be satisfied.)

The $Cost_3[s,r][i][k]$ for transferring the requested data item $\delta[i]$ from machine $M[s]$ to $M[r]$ via link $L[s,r][k]$ using the third heuristic (C3) is:

$$Cost_3[s,r][i][k] = \sum_{j \in D\delta[i,r]} Efp[i,j] / Urgency[i,j].$$

The third criterion takes the weighted priority for a destination and divides it by the urgency for this destination, and then sums over all the destinations with satisfiable requests for data item $\delta[i]$ on a path through machine $M[r]$. This cost is a sum of the weighted priorities of satisfiable requests normalized by the urgency of each request. Note that this heuristic does not use W_E or W_U. This is because the effective priority is divided by the urgency and so W_E divided by W_U acts as a scaling factor that would not affect the relative cost of the requests. That is, for two data items i_1 and i_2 competing for the use of $L[s,r][k]$, the relative value of $Cost_3[s,r][i_1][k]/Cost_3[s,r][i_2][k]$ will be unchanged by including any given W_E to weight the $Efp[i,j]$ factors and any given W_U to weight the $Urgency[i,j]$ factors.

The $Cost_4[s,r][i][k]$ for transferring the requested data item $\delta[i]$ from machine $M[s]$ to $M[r]$ via link $L[s,r][k]$ using the third heuristic (C4) is:

$$Cost_4[s,r][i][k] = -W_E * \sum_{j \in D\delta[i,r]} Efp[i,j] + W_U * \sum_{j \in D\delta[i,r]} -Urgency[i,j].$$

This last criterion sums the weighted priorities of all satisfiable requests for data item $\delta[i]$ on a path through machine $M[r]$ and combines that with the sum of the urgency for those same satisfiable requests. Comparing $Cost_2$ and $Cost_4$, it should be noted that the urgency term for each destination whose shortest path shares an intermediate node $M[r]$ is summed in $Cost_4$, whereas $Cost_2$ simply takes the minimum of the urgency term over this same set of destinations. The benefit of $Cost_4$ is highlighted by the examining the following two data items. The first data item, $\delta[i]$, is requested by four machines that all have identical priorities, and have an A_T that is very close to their deadlines. The second data item, $\delta[j]$, is also requested by four destinations that have the same identical priorities, but only one destination has an A_T that is close to its deadline. $Cost_2$ will be unable to differentiate between these two data requests, but $Cost_4$ will chose to schedule $\delta[i]$ before $\delta[j]$.

All four of these cost criteria are used in conjunction with the partial path heuristic and the full path/one destination heuristic. For the full path/all destinations heuristic, $Cost_1$ is not used because it does not capture the fact that a data item can be sent to multiple destinations.

4. Simulation Study

4.1. Introduction

To perform the simulation study, network topologies and data requests must be generated, values for W_E and W_U must be determined, and other scheduling schemes need to be created to compare to the heuristics discussed in Sections 3.3 through 3.5. Rather than just choosing one network topology and set of data requests, 40 test cases are generated because one cannot accurately reflect the range of possible data requests and network configuration scenarios with one case. The three heuristics are executed using each of these cases and the results are averaged. The properties for data requests and the underlying communication systems are randomly generated with uniform distributions in predefined ranges representing a subset of the systems in a BADD-like environment. The sources and requesting machines for all data items are also generated randomly. The test generation program guarantees that the generated communication system is strongly connected [CoL90], such that there is a path consisting of unidirectional physical transmission links between any pair of machines in both directions.

These randomly generated patterns of data requests and the underlying communication systems are used for three reasons: (1) it is beneficial to obtain cases that can demonstrate the performance of the heuristics over a broad range of conditions; (2) a generally accepted set of data staging benchmark tasks does not exist; and (3) it is not clear what characteristics a "typical" data staging task would exhibit. Determining a representative set of data staging benchmark tasks remains an unresolved challenge in the research field of data staging and is outside the scope of this document. As an example of the communication network structure generated, one sample adjacency matrix is shown in Figure 2. A • at position $adj(i, j)$ in the matrix represents the fact that a link between machine i and machine j exists, a ∘ means there is no such link. The links utilized here are unidirectional. This corresponds to some links being satellite based links. Bidirectional links can be represented as two unidirectional links. Thus, a link from machine i to machine j does not imply that there is a link from machine j to machine i, nor does it disallow such a link. Figure 3 shows the initial times that the links are available before any scheduling of data items occurs. The varying availability of some links correlates to certain links being satellite based and not always available. Other links can be reserved for specific functions at certain times, such as teleconferencing, and unavailable for transferring data items. Not every link shown in Figure 2 is available during the simulation period shown in Figure 3.

Finding optimal solutions to data staging tasks with realistic parameter values are intractable problems. Therefore, it is currently impractical to directly compare the quality of the solutions found by the three heuristics with those found by exhaustive searches in which optimal answers can be obtained by enumerating all the possible schedules of communication steps. Also, to the best of the authors' knowledge, there is no other work presented in the open literature that addresses the data staging problem and presents a heuristic for solving it (based on a similar underlying model). Thus, there is no other heuristic for solving the same problem with which to make a direct comparison of the heuristics presented in this document. To aid in the evaluation of these heuristics, two lower bounds (see Subsection 4.2), and two upper bounds (see Subsection 4.4) on the performance of the heuristics are provided.

4.2. Random Search Based Scheduling

The performance of the three heuristics presented here are compared with two random search based scheduling procedures. The only difference between the first random procedure and the partial path heuristic is that, instead of choosing a valid communication step using a cost function as discussed for the partial path heuristic, the Dijkstra random heuristic (shown as random_dijkstra in Figures 4 through 9) randomly chooses an arbitrary valid communication step to schedule. This heuristic is used to show the importance of using a cost criterion for decision making.

The second random-search based scheduling procedure performs Dijkstra once for each requested data item, assuming it is the only requested item in the network. Then the paths through the network are scheduled for each data item, finishing $\delta[i]$ before $\delta[i+1]$ (where the ordering of the data items is arbitrary). If a conflict arises, e.g., the link a transfer is attempting to schedule is no longer available, the request is dropped and not satisfied. This approach is referred to as single Dijkstra random (shown as single_dij_random in Figures 4 through 9) because Dijkstra's algorithm is only executed once for each data item. This random based method is used to present that executing Dijkstra's algorithm again, with updated communication system information is advantageous.

4.3. Parameters Used in Experiments

Creating the properties of a network structure that are expected to occur in the field is a difficult endeavor. The parameters used were chosen to reflect a representative subset of a BADD-like environment. The number of machines in the communication system is between ten and twelve. Each machine has between 10MB to 20GB memory storage capacity. The outbound degree of a machine $M[i]$ (i.e., the number of machines that $M[i]$ can transfer data items to directly through physical transmission links) is between four and seven. There are at most two physical unidirectional transmission links between any two machines (there can be none). The adjacency matrix is created by selecting each machine in the network and randomly determining the outbound degree of the machine to be between the bounds mentioned above. Once the outbound degree is chosen, the end machines for the links are randomly generated. The network creation software makes sure that a link does not originate and end at the same machine.

The total number of data requests is 20 to 40 times the number of machines in the system. The sources and destinations for a data item are randomly selected from the set of machines in the system such that: (1) there are at most five sources, (2) there are at most five destinations, and (3) a destination for a data item is not also a source of the same data item. Each data item size ranges from 10KB to 100MB. The simulations that were performed utilized two different priority weightings. The first used a weighting of 1, 5, and 10 for low, medium, and high priority requests, and the second used a weighting of 1, 10, and 100 for low, medium, and high priority requests. Each data item a destination requests has an associated priority; therefore, two destinations that request the same data item may have differing priorities for that data item.

The bandwidth of each physical transmission link is between 10Kbit/sec and 1.5Mbit/sec. The link availability times were generated as follows. First, a duration for a particular virtual communication link between two machines was chosen from the set {30 minutes, 1 hour, 2 hours, 4 hours}. The percentage of the day (24 hours) that this link is available is then chosen between 50 and 100 percent of the day, in increments of 10 percent. The number of virtual communication links is then determined by taking the time the link is available during the day (percentage available * 24 hours) and dividing by the duration of the virtual link chosen above. The unavailable time between virtual links is randomly chosen such that: (1) no two virtual links for the same physical link overlap in time, (2) the percentage of the day the physical link is available is as chosen above, and (3) the number of virtual links is as calculated above. The starting time for a data item is sometime between 0 and 60 minutes (0 signifying some start time of the scheduling period, such as midnight or 6 a.m.). The deadline for the data item is 15 to 60 minutes after the available time. Thus, the effective duration of the simulation is two hours. The time duration parameter for garbage collection, γ, was set to six minutes. Thus, a particular machine $M[r]$ will keep a data item in its local memory for six minutes after all destinations, whose shortest path used $M[r]$, have received $\delta[i]$. These parameters were used as they capture the information about the network that is necessary to show the functionality of the heuristic over a variety of network configurations.

Let the E-U ratio be W_E/W_U. As shown by the cost functions introduced in Subsection 4.4, the E-U ratio may affect the performance of the cost criterion ($Cost_1$, $Cost_2$, and $Cost_4$). Figure 4 shows the performance of the best heuristic/cost-criterion pairings examined. It can be seen from the figure how the E-U ratio affects the performance of the heuristics. All data points are the average of 40 randomly generated test cases.

4.4. Evaluation of Simulations

The results shown in Figures 4 through 7 are for the 1, 10, 100 weighting scheme. The results of the 1, 5, 10 weighting are similar and are not shown here. In the interval from -3 to 5, the horizontal axis contains the log_{10} of the E-U ratio. The points inf and $-inf$ are the two extremes where inf only considers the effective priority term, while $-inf$ only considers the urgency term.

Figure 4 shows the performance of the best cost criterion for each of the heuristics, which happens to be $Cost_4$, and the bounds used in the experiments. The upper bound is the total weighted sum of the priorities of all requests in the system. The loose upper bound represents those requests that could be satisfied if they were the only requests in the system (shown as upper bound and possible_satisfy in Figure 4). Note that this line is not equal to the upper bound, because some requests cannot be satisfied due to lack of link bandwidth and/or machine storage. The lower bounds show the single Dijkstra random and the Dijkstra random heuristics performance. The single Dijkstra random performs poorly compared to the non-random heuristics and the Dijkstra random heuristic.

The performance of the partial path heuristic is shown in Figure 5, the full path/one destination heuristic in Figure 6, and the full path/all destinations heuristic in Figure 7. These graphs highlight the performance of the four cost criteria for each of the heuristics implemented. For the best performing E-U ratio, for each of the heuristic/cost-criterion pairs, the average number of links a satisfied data item traverses from a source to a destination was measured in the range 1.5 to 1.6. All potentially satisfiable data items have an average path length that is also in the above range. The maximum path length taken by a satisfied data item was measured as nine.

The various heuristics also have differing execution times. Figure 8 shows how the average execution time varies for the three heuristics examined here as well as the Dijkstra/random heuristic. The full path/all destination heuristic requires the least amount of time to achieve its best performance, while Dijkstra/random requires the most time. The random Dijkstra heuristic execution time is larger than the cost-based heuristics because the random Dijkstra heuristic will transfer data items that cannot possibly satisfy the associated deadlines, whereas the cost-based heuristics will not transfer such data items. The best performing heuristic, as can be seen in Figure 4, is the full path/one destination heuristic which has an execution time between these two extremes.

The last figure, Figure 9, shows the comparison between the two priority weighting schemes for the best combination of heuristic and cost criterion. It can be seen that the 1, 10, 100 weighting satisfies more higher priority requests and fewer medium and low priority requests than the 1, 5, 10 weighting scheme. This is as desired; if a high priority request being satisfied is equivalent to satisfying ten medium priority requests, one would satisfy more of the high priority requests than when the high priority request being satisfied is equivalent to satisfying two medium priority requests. The combination of $Cost_1$, the partial path heuristic, and an E-U ratio of inf has the property that all high priority requests would be considered before any medium priority requests, which would be considered before any low priority requests. In this case, it was expected that more high priority requests would be satisfied than shown by the results in Figure 9 because these high priority requests would have the first chance of using the network. This was not shown in the results. Instead fewer high priority requests were satisfied. This is attributed to the fact that when the local cost criterion has a tie, the first data item that resulted in that cost is transferred. This transfer could require large amounts of bandwidth and delay other more urgent high priority requests from being transferred.

5. Conclusions

Data staging is an important data management issue for information systems. It addresses the issues of distributing and storing over numerous geographically dispersed locations both repository data and continually generated data. When certain data with their corresponding priorities need to be collected together at a site with limited storage capacities in a timely fashion, a heuristic must be devised to schedule the necessary communication steps efficiently. Because each heuristic/cost-criterion pair has advantages, the pair that performs best may differ depending on the way in which system parameters are generated (i.e., the underlying assumptions about the actual environment where the scheduler heuristic/cost-criterion pair will be deployed).

Three heuristics and four cost criteria have been introduced to solve a simplified version of the data staging problem. Ten of the twelve possible combinations have been first explored in this paper. The eleventh was explored in [TaT98] utilizing a simpler network model. The network model used here has been modified from [TaT98] to allow a more realistic version of garbage collection to be implemented and to have various network parameters more closely match a BADD-like environment. The twelfth heuristic/cost-criterion (the full path/all destinations and $Cost_1$) did not make sense and was not examined in this research.

The performance of eleven heuristic/cost-criterion pairs were shown, and compared to upper and lower bounds. Each heuristic and cost criterion had advantages. The results presented show that for the system parameters considered, the combination of the $Cost_4$ and the full path/one destination heuristic performed the best, when using the metric of weighted sum of priorities satisfied. If one was more concerned with execution time, the full path/all destinations heuristic results in a comparable weighted sum of priorities satisfied with a faster execution time. Future work will explore varying the congestion of the network and additional priority weighting schemes to examine how the heuristics perform.

6. Acknowledgments

The author's would like to thank Bob Beaton, Surja Chatterjea, Bob Douglas, and Joe Rockmore for their comments and suggestions.

7. References

[AcZ93] S. Acharya and S. B. Zdonik, "An efficient scheme for dynamic data replication," Tech. Report CS-93-43, Dept. of Computer Science, Brown Univ., 1993, 25 pp.

[BaB97] M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm, "Enhancing the web's infrastructure: From caching to replication," *IEEE Internet Computing*, Vol. 1, No. 2, March-April 1997, pp. 18-27.

[Bes97] A. Bestavros, "WWW traffic reduction and load balancing through server-based caching," *IEEE Concurrency*, Vol. 5, No. 1, January-March 1997, pp. 56-67.

[ChD81] R. Chandrasekaran and A. Dauchety, "Location on tree networks: P-centre and n-dispersion problems," *Mathematics of Operations Research*, Vol. 6, No. 1, February 1981, pp. 50-57.

[CoL90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.

[CoN80] G. Cornuejols, G. L. Nemhauser, and L. A. Wolsey, "Worst-case and probabilistic analysis of algorithms for a location problem," *Operations Research*, Vol. 28, No. 4, July-August 1980, pp. 847-858.

[DaH93] P. Danzig, R. Hall, and M. Schwartz, "A case for caching file objects inside internetworks," Tech. Report CU-CS-642-93, Computer Science Dept., Univ. of Colorado, 1993, 15 pp.

[HuM89] A. P. Hurter and J. S. Martinich, *Facility Location and The Theory of Production*, Kluwer Academic Publishers, Norwell, MA, 1989.

[JoL95] P. C. Jones, T. J. Lowe, G. Muller, N. Xu, Y. Ye, and J. L. Zydiak, "Specially structured uncapacitated facility location problem," *Operations Research*, Vol. 43, No. 4, July-August 1995, pp. 661-669.

[LeB97] M. J. Lemanski and J. C. Benton, *Simulation for SmartNet Scheduling of Asynchronous Transfer Mode Virtual Channels,* Master of Science Thesis, Dept. of Computer Science, Naval Postgraduate School, June 1997 (Advisor: D. Hensgen).

[MoC84] I. D. Moon and S. S. Chaudhry, "An analysis of network location problems with distance constraints," *Management Science*, Vol. 30, No. 3, March 1984, pp. 290-307.

[Roc96] A. J. Rockmore, "BADD functional description," Internal DARPA Memo, February 1996.

[Sma96] SmartNet/Heterogeneous Computing Team, "BC2A/TACITUS/BADD integration plan," Internal Report, August 1996.

[Shi77] D. R. Shier, "A min-max theorem for p-center problems on a tree," *Transportation Science*, Vol. 11, No. 3, August 1977, pp. 243-252.

[TaS97] M. Tan, H. J. Siegel, J. K. Antonio, and Y. A. Li, "Minimizing the application execution time through scheduling of subtasks and communication traffic in a heterogeneous computing system," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 8, No. 8, August 1997, pp. 857-871.

[TaT98] M. Tan, M. D. Theys, H. J. Siegel, N. B. Beck, and M. Jurczyk, "A mathematical model, heuristic, and simulation study for a basic data staging problem in a heterogeneous networking environment," *Proceedings of the 7th Heterogeneous Computing Workshop (HCW'98)* , Mar. 1998, pp. 115-129.

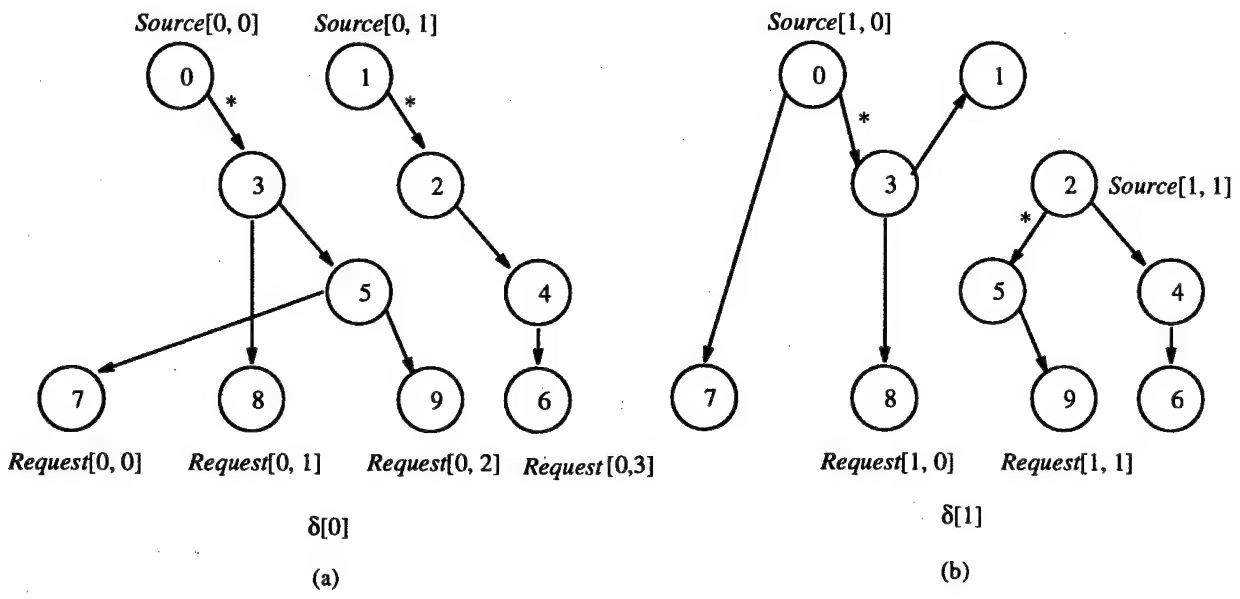

Figure 1: An example communication system that requests (a)$\delta[0]$ and (b)$\delta[1]$. $Source[i, j]$ denotes the j-th initial source location of the data item $\delta[i]$. $Request[i, j]$ denotes the machine from which the j-th request for data item $\delta[i]$ originates.

$$\begin{pmatrix}
\circ & \bullet & \circ & \bullet & \circ & \bullet & \circ & \circ & \bullet & \bullet \\
\circ & \circ & \bullet & \circ & \bullet & \bullet & \circ & \bullet & \bullet & \circ \\
\bullet & \circ & \circ & \circ & \bullet & \bullet & \circ & \circ & \circ & \bullet \\
\circ & \bullet & \bullet & \circ & \bullet & \circ & \circ & \bullet & \bullet & \bullet \\
\bullet & \circ & \bullet & \circ & \circ & \circ & \bullet & \circ & \bullet & \bullet \\
\circ & \bullet & \bullet & \bullet & \circ & \circ & \circ & \bullet & \bullet & \circ \\
\bullet & \bullet & \circ & \circ & \bullet & \bullet & \circ & \circ & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \circ & \circ & \circ & \circ & \bullet & \bullet \\
\circ & \circ & \bullet & \bullet & \circ & \circ & \bullet & \circ & \circ & \circ \\
\circ & \circ & \bullet & \bullet & \bullet & \bullet & \circ & \circ & \circ & \circ
\end{pmatrix}$$

Figure 2: An example adjacency matrix for the networks used.

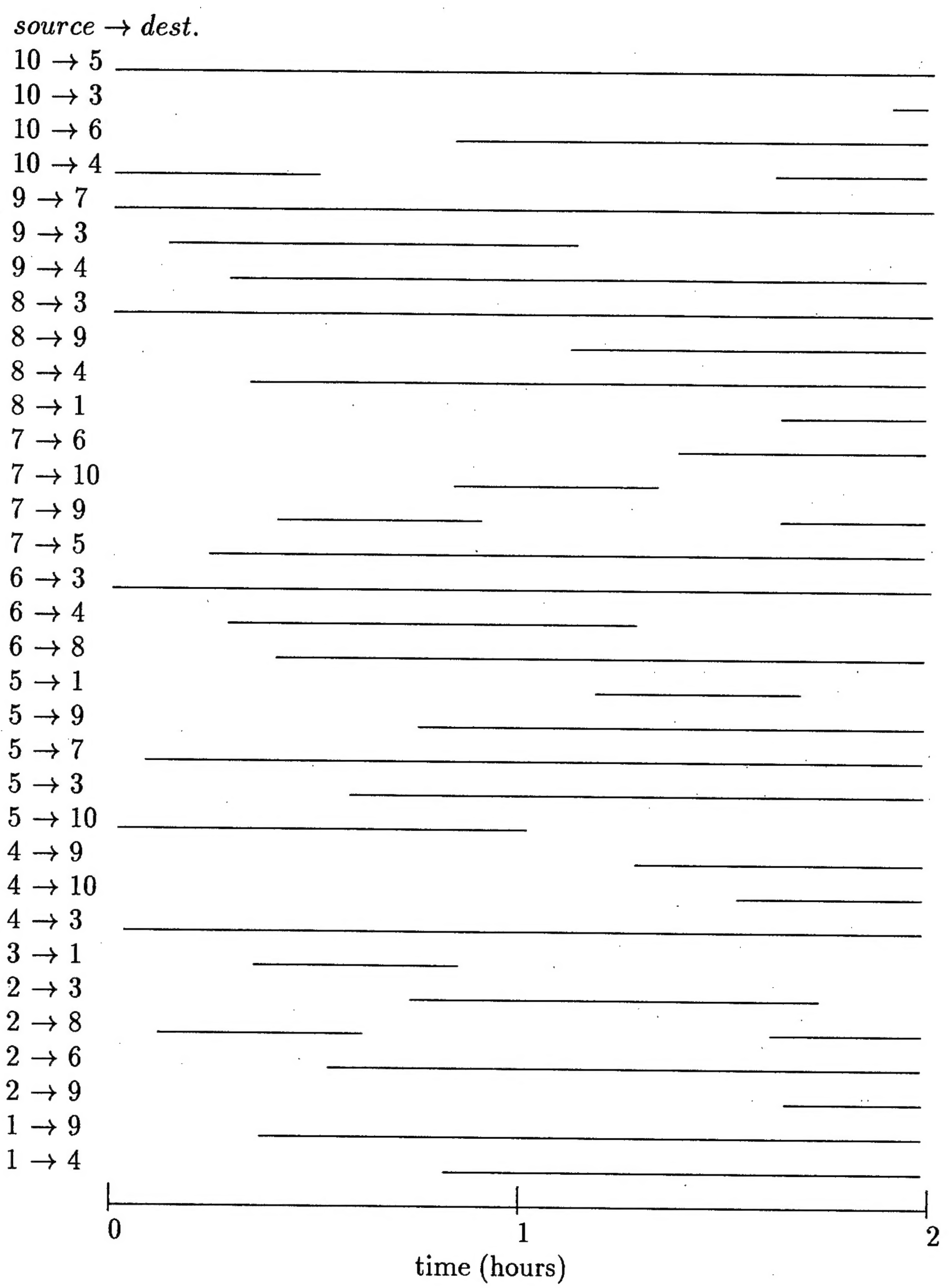

Figure 3: An example of the initial link availability for the networks used.

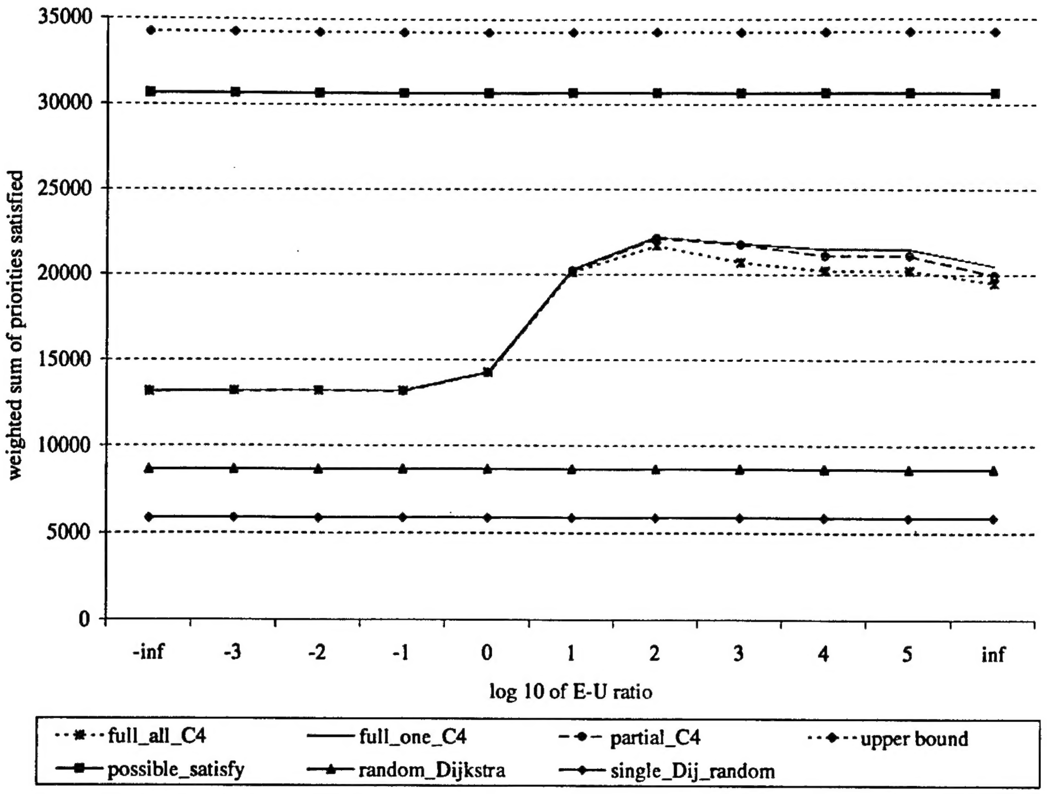

Figure 4: Comparison of the heuristics best cost criterion performance.

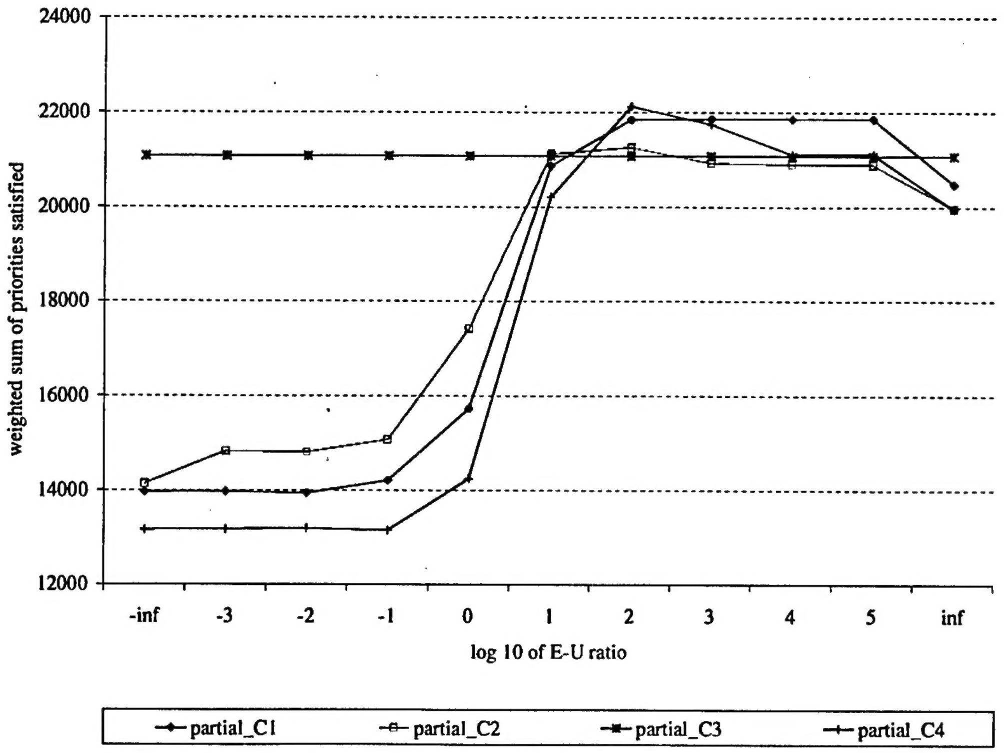

Figure 5: The partial path heuristic results.

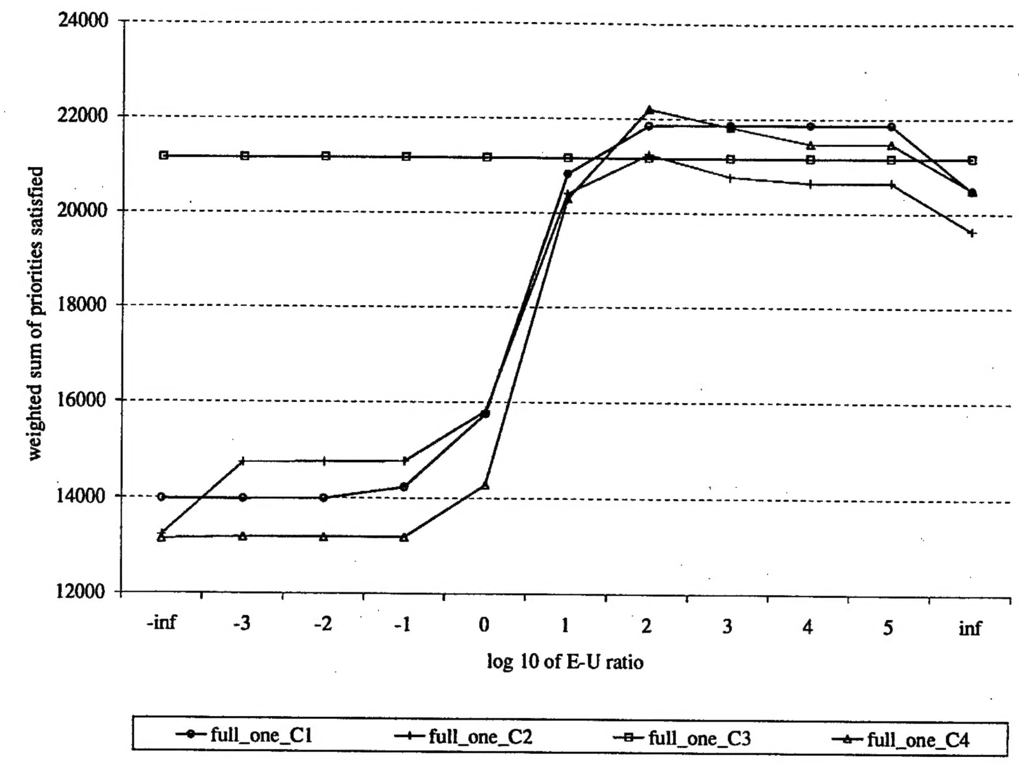

Figure 6: The full path/one destination heuristic results.

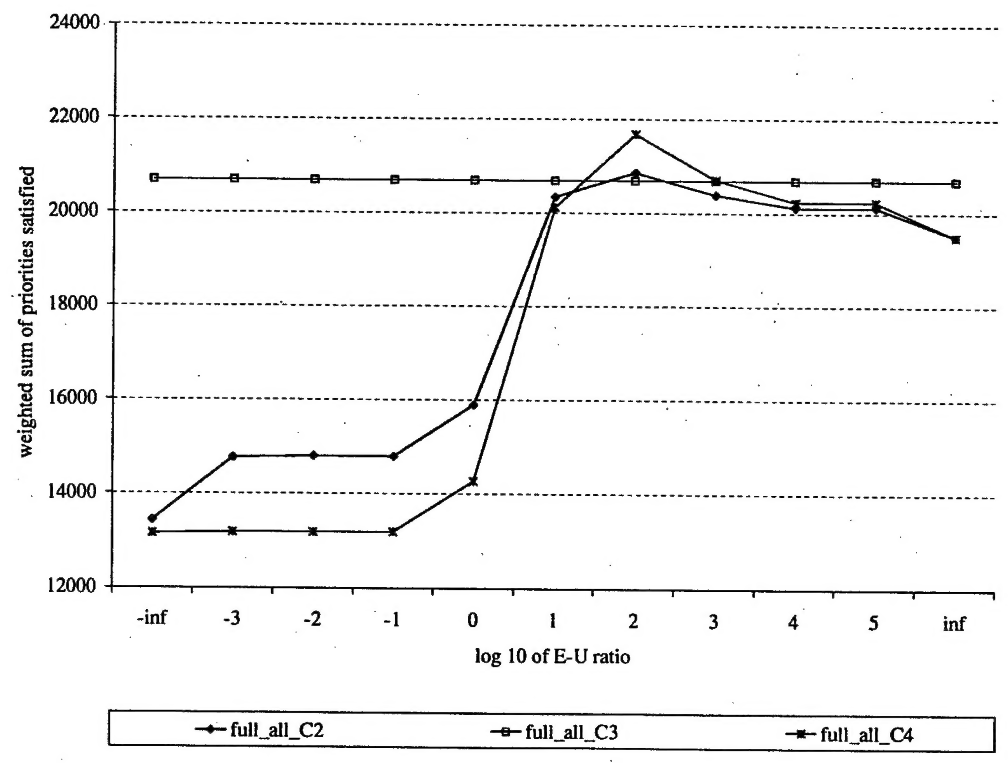

Figure 7: The full path/all destinations heuristic results.

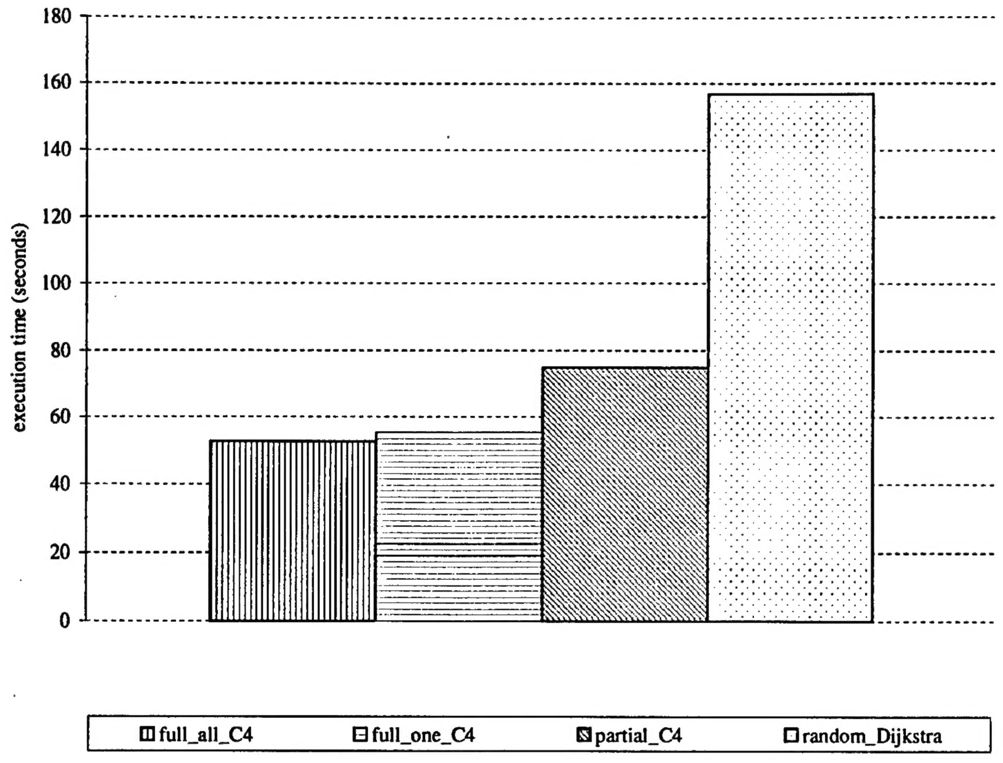

Figure 8: A comparison of average execution times.

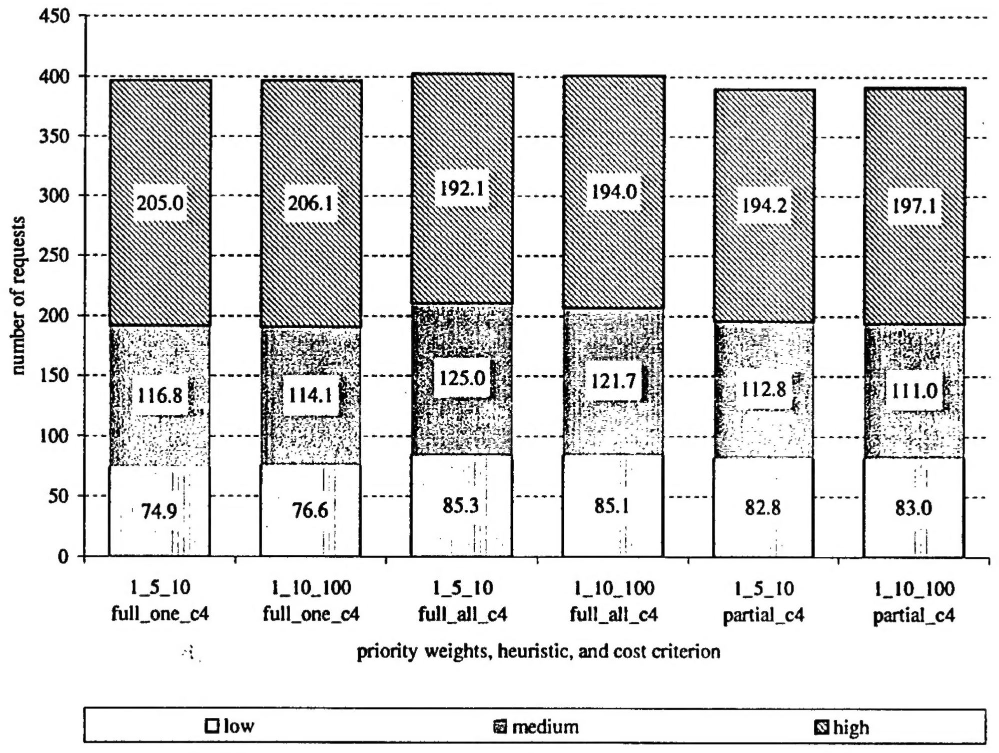

Figure 9: A comparison of the best heuristic/cost criterion combination using different priority weighting terms.